J2EE vs. Microsoft® .NET Market Bulletin

Topic: Portability

June 2001

One of the biggest promises of Java 2 Enterprise Edition (J2EE) -based applications is portability. Portability means the ability to either "Write Once Run Anywhere" or easily be able to move the application from one hardware platform to another as computing needs change. This seems logical, because after all, J2EE is a public specification. In fact, there are over 20 different vendors offering J2EE-based applications, some of which are also "certified" to be "compliant" with a standard set of tests. On the surface, it can even appear that choosing the Microsoft way means a significant sacrifice in portability over a J2EE based solution. In this bulletin, we discuss what kind of portability is actually available today for customers who choose J2EE.

THE J2EE SPECIFICATION IS INCOMPLETE

One of the main problems with the current J2EE specification and the related server-side Java technologies is that they do not describe a complete application platform – key functionality is missing from the specification, causing vendors to create functionality on their own in an inconsistent way.

See Giga Group's report, titled, What is an Application Platform? at http://www.gigaweb.com/Content/GIB/rib-102000-00053.html for details. As a result, vendors providing a usable application platform will not be bound by "standards". Specific examples:

- J2EE does not specify enterprise application integration; business process management; or workflow. These are critically important areas to consider when assembling an enterprise software application platform, and each vendor handles them differently.
- J2EE does not prescribe development tools. Each vendor provides an integrated toolset that works best with the vendor's own application server. While vendors sometimes claim that different tools can be used with different app servers, in fact there is tight coupling between the tool and the application server.
- J2EE does not prescribe management interfaces. Each Enterprise Java Bean application server vendor employs different management and administrative interfaces – some of them with a graphical user interface, and some of them exposed as text "property files" or "ini files". Each of the products exposes a completely proprietary and independent administrative interface.
- J2EE does not discuss or prescribe how to employ XML; for example, while a Java/XML parser is included in the J2EE reference implementation,
    - Sun does not describe how to translate between a result set from a database operation (typically Java Database Connectivity or JDBC) into an XML document; this is largely done either (i) manually, (ii) in a parser-specific

manner, or (iii) in a database-specific manner (e.g., using stored procedures supplied by the database vendor).

  - Sun does not describe how to convert between an entity object and an XML document. Again, this must be done in an extension (often proprietary) provided by the vendor or manually with additional programming.
  - There are numerous Java XML parsers available; none are standard, any could be used within J2EE.
- In addition, at a base level, the J2EE does not specify a common way to implement shared HTTP sessions, which allows fail over to occur, load balancing, and application logging.

Every one of these non-specified areas is an opportunity for a vendor to implement them in a proprietary way.

- Finally, the J2EE sample application, J2EE PetShop, has different versions for different vendor products. See http://java.sun.com/j2ee/download.html . There is a distinct version for each different Application Server Vendor. One would reasonably expect that PetShop would be able to run unaltered on any platform, based on the claim of portability.

"J2EE-COMPLIANT" DOES NOT MEAN AVOIDING VENDOR LOCK-IN

Because vendors claiming J2EE support implement outside-the-spec features in a non-standard fashion, customer applications built on such products will inherently be closely dependent upon those products. A leading analyst, after reviewing the JavaOne 2001 announcements, stated, "*We interpret the message from JavaOne as follows:*

Users should not expect 100 percent compatibility among different Java vendors' platforms, any more than they should expect that from different versions of UNIX. ***And the gulf between different Java solutions will widen****.*

Another analyst stated in October 2000, "*Clients can lose the portability benefits of EJB by getting locked into some other part of the application platform*." Yet another analyst states, "*The primary drawback to new generation toolsets (for J2EE) will inevitably be the degree of vendor lock-in*." Also, "*…[the Java Platform's] proprietary infrastructure services (e.g., workflow, messaging, persistence, legacy integration) will create substantial barriers to application portability."*

IBM publishes a 274-page document describing how to port Java applications from a competitor's application server to WebSphere. Does this sound like write-once run-anywhere™ ?

Finally, Roger Sessions, a well-known middleware expert said, *"The idea that J2EE in any way supports vendor independence is a serious misunderstanding. Sun's Enterprise JavaBeans specification (part of J2EE) is quite clear on this point, saying that the only way to achieve vendor independence is by adhering to the minimum technology specified by Sun… Even the most basic commerce application will require many features that exceed the Sun specifications."*

Vendors will naturally try to create competitive advantage with their J2EE products, which will by definition result in products that work best on their platforms, and aren't compatible with rivals' offerings. This is often referred to as "vendor lock-in."

Microsoft does not believe that vendor lock-in is necessarily bad business. What we object to is the widespread perception that J2EE vendors will somehow allow customers to avoid vendor lock-in. Our position is that avoiding vendor lock-in by employing the least-common-denominator approach implies a cost that is too great – lost performance, lost optimizations. Customers should establish strategic relationships with those vendors they believe are best positioned to provide a viable, low-cost, high-performance application platform, now and in the future. These relationships – "vendor commitment" or "vendor lock-in", depending on your perspective - represent a key opportunity for savvy IT organizations as they strive to build an agile, responsive IT organization.

Where to go for more information

Independent Discussion of J2EE vs. .NET by Roger Sessions

http://www.objectwatch.com/issue_24.htm

GigaGroup report

http://www.gigaweb.com/Content/GIB/rib-102000-00053.html)

GartnerGroup: Research Note on Java, October 2000 – by Mark Driver, document id# M-11-9289

IBM's 274-page porting document

http://publib-b.boulder.ibm.com/Redbooks.nsf/RedbookAbstracts/sg245956.html